LARGE SCALE MACHINE LEARNING SYSTEMS AND METHODS

RELATED APPLICATION

[0001] This application is a continuation-in-part of U.S. Patent Application, Serial No. 10/706,991 (docket no. 0026-0057), filed November 14, 2003, and entitled "RANKING DOCUMENTS BASED ON LARGE DATA SETS," the disclosure of which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The present invention relates generally to classification systems and, more particularly, to systems and methods for applying machine learning to various large data sets to generate a classification model.

Description of Related Art

[0003] Classification models have been used to classify a variety of elements. The classification models are built from a set of training data that usually includes examples or records, each having multiple attributes or features. The objective of classification is to analyze the training data and develop an accurate model using the features present in the training data. The model is then used to classify future data for which the classification is unknown. Several classification systems have been proposed over the years, including systems based on neural networks, statistical models, decision trees, and genetic models.

[0004] One problem associated with existing classification systems has to do with the volume of training data that they are capable of handling. Existing classification systems can only efficiently handle small quantities of training data. They struggle to deal with large quantities of data, such as more than one hundred thousand features.

[0005] Accordingly, there is a need for systems and methods that are capable of generating a classification model from a large data set.

SUMMARY OF THE INVENTION

[0006] Systems and methods, consistent with the principles of the invention, apply machine learning to large data sets to generate a classification model.

[0007] In accordance with one aspect consistent with the principles of the invention, a system for generating a model is provided. The system may include multiple nodes. At least one of the nodes is configured to select a candidate condition, request statistics associated with the candidate condition from other ones of the nodes, receive the requested statistics from the other nodes, form a rule based, at least in part, on the candidate condition and the requested statistics, and selectively add the rule to the model.

[0008] According to another aspect, a system for generating a model is provided. The system may form candidate conditions and generate statistics associated with the candidate conditions. The system may also form rules based, at least in part, on the candidate conditions and the generated statistics and selectively add the rules to the model.

[0009] According to yet another aspect, a method for generating a model in a system that includes multiple nodes is provided. The method may include generating candidate conditions, distributing the candidate conditions to the nodes, and generating statistics regarding the candidate conditions. The method may also include collecting the statistics for each of the candidate conditions at one of the nodes, generating rules based, at least in part, on the statistics and the candidate conditions, and selectively adding the rules to the model.

[0010] According to a further aspect, a system for generating a model is provided. The system may generate new conditions and distribute the new conditions to a set of nodes. Each of the nodes may generate statistics regarding the new conditions. The system may generate new rules based, at least in part, on the statistics and the new conditions and add at least one of the new rules to the model.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

[0012] Fig. 1 is a diagram of an exemplary model generation system according to an implementation consistent with the principles of the invention;

[0013] Fig. 2 is an exemplary diagram of a node of Fig. 1 according to an implementation consistent with the principles of the invention;

[0014] Fig. 3 is a flowchart of exemplary processing for generating a model according to a first implementation consistent with the principles of the invention; and

[0015] Fig. 4 is a flowchart of exemplary processing for generating a model according to a second implementation consistent with the principles of the invention; and

[0016] Fig. 5 is a flowchart of exemplary processing for generating a model according to a third implementation consistent with the principles of the invention.

DETAILED DESCRIPTION

[0017] The following detailed description of the invention refers to the accompanying drawings. The same reference numbers in different drawings may identify the same or similar elements. Also, the following detailed description does not limit the invention.

[0018] Systems and methods consistent with the principles of the invention may apply machine learning to large data sets, such as data sets including over one hundred thousand features and/or one million instances. The systems and methods may be capable of processing a large data set in a reasonable amount of time to generate a classification model.

[0019] Different models may be generated for use in different contexts. For example, in an exemplary e-mail context, a model may be generated to classify e-mail as either spam or normal (non-spam) e-mail. In an exemplary advertisement context, a model may be generated to estimate the probability that a user will click on a particular advertisement. In an exemplary document ranking context, a model may be generated in connection with a search to estimate the probability that a user will find a particular search result relevant. Other models may be generated in other contexts where a large number of data items exist as training data to train the model.

EXEMPLARY MODEL GENERATION SYSTEM

[0020] Fig. 1 is an exemplary diagram of a model generation system 100 consistent with the principles of the invention. System 100 may include nodes 110-1 through 110-N (collectively referred to as nodes 110) optionally connected to a repository 120 via a network 130. Network 130 may include a local area network (LAN), a wide area network (WAN), a telephone network, such as the Public Switched Telephone Network (PSTN), an intranet, the Internet, a memory device, another type of network, or a combination of networks.

[0021] Repository 120 may include one or more logical or physical memory devices that may store a large data set (e.g., potentially over one million instances and/or one hundred thousand features) that may be used, as described in more detail below, to create and train a model. In the description to follow, the data set will be described in the exemplary e-mail context and, thus, data items relating to e-mail may be described. One of ordinary skill in the art would understand how to extend the description to other contexts.

[0022] In the exemplary e-mail context, the data set in repository 120 will be called "D." D may include multiple elements "d," called instances. Each instance d may include a set of features "X" and a label "Y." In one implementation, the label Y may be a boolean value (e.g., "spam" or "non-spam"), which may be called y_0 and y_1. In another implementation, the label Y may be a discrete value (e.g., values corresponding to categories of labels).

[0023] A feature X may be an aspect of the domain (e.g., the e-mail domain) that may be useful to determine the label (e.g., "the number of exclamation points in the message" or "whether the word 'free' appears in the message"). In one implementation, each feature X may include a boolean value (e.g., a value of zero or one based on whether the word "free" appears in the message). In another implementation, each feature X may include a discrete value (e.g., a value based, at least in part, on the number of exclamation points in the message). In yet another implementation, each feature X may include a real value (e.g., the time of day a message was sent). An instance d may be written as: $d = (x_1, x_2, x_3, \ldots, x_m, y)$, where x_i is the value of the i-th feature X_i and y is the value of the label.

[0024] Repository 120 could potentially store millions of distinct features. For efficiency, an instance d may be encoded using a sparse representation: if x_i is zero, then its value is not stored for d. For example, assume that X_2 is the feature "does the word 'free' appear in the message."

For a particular instance d, if the word "free" does not appear in the message, then x_2 is not stored for d.

[0025] Nodes 110 may include entities. An entity may be defined as a device, such as a personal computer, a wireless telephone, a personal digital assistant (PDA), a lap top, or another type of computation or communication device, a thread or process running on one of these devices, and/or an object executable by one of these device.

[0026] Each of nodes 110 may be responsible for a subset of instances. In one implementation, nodes 110 obtain their subset of instances from repository 120 when needed. In another implementation, each of nodes 110 may optionally store a copy of its subset of instances in a local memory 115. In this case, nodes 110 may retrieve their copy from repository 120. In yet another implementation, each of nodes 110 may store its subset of instances in local memory 115 and system 100 may include no repository 120.

[0027] Fig. 2 is an exemplary diagram of a node 110 according to an implementation consistent with the principles of the invention. Node 110 may include a bus 210, a processor 220, a main memory 230, a read only memory (ROM) 240, a storage device 250, one or more input devices 260, one or more output devices 270, and a communication interface 280. Bus 210 may include one or more conductors that permit communication among the components of node 110.

[0028] Processor 220 may include any type of conventional processor or microprocessor that interprets and executes instructions. Main memory 230 may include a random access memory (RAM) or another type of dynamic storage device that stores information and instructions for execution by processor 220. ROM 240 may include a conventional ROM device or another type of static storage device that stores static information and instructions for use by processor 220.

Storage device 250 may include a magnetic and/or optical recording medium and its corresponding drive.

[0029] Input device(s) 260 may include one or more conventional mechanisms that permit an operator to input information to node 110, such as a keyboard, a mouse, a pen, voice recognition and/or biometric mechanisms, etc. Output device(s) 270 may include one or more conventional mechanisms that output information to the operator, including a display, a printer, a speaker, etc. Communication interface 280 may include any transceiver-like mechanism that enables node 110 to communicate with other nodes 110 and/or repository 120.

[0030] As will be described in detail below, node 110, consistent with the principles of the invention, may perform certain operations relating to model generation. Node 110 may perform these operations in response to processor 220 executing software instructions contained in a computer-readable medium, such as memory 230. A computer-readable medium may be defined as one or more physical or logical memory devices and/or carrier waves.

[0031] The software instructions may be read into memory 230 from another computer-readable medium, such as data storage device 250, or from another device via communication interface 280. The software instructions contained in memory 230 causes processor 220 to perform processes that will be described later. Alternatively, hardwired circuitry may be used in place of or in combination with software instructions to implement processes consistent with the principles of the invention. Thus, implementations consistent with the principles of the invention are not limited to any specific combination of hardware circuitry and software.

EXEMPLARY MODEL GENERATION PROCESSING

[0032] To facilitate generation of the model, a prior probability of the label for each instance may be determined: $P(Y \mid Z)$. This prior probability can be based on Z, which may include one or more values that differ based on the particular context in which the model is used. Typically, Z may be real valued and dense (i.e., it does not include many zero entries for many of the instances). In the e-mail context, each e-mail may be evaluated using a common spam detection program that gives each e-mail a score (e.g., Spam Assassin). The output of the spam detection program may be used as the prior probability that the e-mail is spam.

[0033] A set of instances based on the same or a different set of instances as in repository 120 or memory 115 may be used as "training data" D. For each instance d in the training data D, its features (X_0, X_1, . . . , X_m) may be extracted. For example, X_0 may be the feature corresponding to "the message contains the word 'free.'" In this implementation, the feature X_0 may include a boolean value, such that if "free" appears in the message, then x_0 is one, otherwise x_0 is zero. In other implementations, the features may include discrete values. It may be assumed that many of the features will have values of zero. Accordingly, a sparse representation for the features of each instance may be used. In this case, each instance may store only features that have non-zero values.

[0034] As will be explained later, it may be beneficial to quickly obtain statistics for the instances that contain particular features. To facilitate fast identification of correspondence between features and instances, a feature-to-instance index may be generated in some implementations to link features to the instances in which they are included. For example, for a given feature X, the set of instances that contain that feature may be listed. The list of instances for a feature X is called the "hitlist for feature X." Thereafter, given a set of features X_0, . . . , X_m, the set of instances that contain those features can be determined by intersecting the hitlist for each of the features X_0, . . . , X_m.

[0035] A "condition" C is a conjunction of features and possibly their complements. For example, a condition that includes two features is: "the message contains the word 'free'" and "the domain of the sender is "hotmail.com," and a condition that includes a feature and a complement of a feature is: "the message contains the word 'free'" and "the domain of the sender is not 'netscape.net.'" For any instance d_i, the value of its features may determine the set of conditions C that apply. A "rule" is a condition C_i and a weight w_i, represented as (C_i, w_i). The model M may include a set of rules and a prior probability of the label.

[0036] Based, at least in part, on this information, a function may be created that maps conditions to a probability of the label: $P(Y \mid C_1, \ldots, C_n, Z)$. The posterior probability of the label given a set of conditions, $P(Y \mid C_1, \ldots, C_n, Z)$, may be determined using the function:

$$\mathrm{Log}\ \{ P(Y = y_0 \mid C_1, \ldots, C_n, Z) / P(Y = y_1 \mid C_1, \ldots, C_n, Z) \}$$
$$= \mathrm{Sum}_i\ \{ -w_i\, I(C_i) \} + \mathrm{Log}\ \{ P(Y = y_0 \mid Z) / P(Y = y_1 \mid Z) \}, \qquad \text{(Eqn. 1)}$$

where $I(C_i) = 0$ if C_i = false, and $I(C_i) = 1$ if C_i = true.

[0037] Thereafter, given a new instance d and a model M, the posterior probability of the label may be determined by: (1) extracting the features from the instance, (2) determining which rules apply, and (3) combining the weight of each rule with the prior probability for instance d. Therefore, the goal is to generate a good model. To generate a good model, the following information may be beneficial: the set of conditions $C_1, \ldots, C_n$, and the values of weights $w_1, \ldots, w_n$.

[0038] Fig. 3 is a flowchart of exemplary processing for generating a model according to a first implementation consistent with the principles of the invention. This processing may be performed by a combination of nodes 110. Each node 110 may include a copy of the model M and a subset of instances with a current probability of $Y = y_1$ for each instance. Each node 110 may build its own feature-to-instance index for its subset of instances.

[0039] Processing may begin with an empty model M that includes the prior probability of the label. A node 110 may select a candidate condition C to be tested (act 310). It may be possible for multiple nodes 110, or all of nodes 110, to concurrently select candidate conditions. In one implementation, nodes 110 may select candidate conditions from the instances in training data D. For example, for each instance, combinations of features that are present in that instance (or complements of these features) may be chosen as candidate conditions. In another implementation, random sets of conditions may be selected as candidate conditions. In yet another implementation, single feature conditions may be considered as candidate conditions. In a further implementation, existing conditions in the model M may be augmented by adding extra features and these augmented conditions may be considered as candidate conditions. In yet other implementations, candidate conditions may be selected in other ways.

[0040] Node 110 may then estimate a weight w for condition C (act 320). Assume that condition C includes three features: X_1 and X_5 and X_{10}. In order to find the set of instances that satisfy condition C, node 110 may use its feature-to-instance index. Given the set of instances that satisfy the condition C, node 110 may gather statistics regarding these instances. If the label of instance d is y[d] and instance d satisfies conditions $C_1, \ldots, C_k$, then node 110 may determine first and second derivatives of:

$$\mathrm{Sum}_d \{ \mathrm{Log}\, P(Y= y[d] \mid C_1, \ldots, C_k, C) \} - \mathrm{Sum}_d \{ \mathrm{Log}\, P(Y = y[d] \mid C_1, \ldots, C_k) \}$$

$$= \mathrm{Sum}_d \{ \mathrm{Log}\, P(Y = y[d] \mid C_1, \ldots, C_k, C) - \mathrm{Log}\, P(Y = y[d] \mid C_1, \ldots, C_k) \}, \quad \text{(Eqn. 2)}$$

where $P(y[d] \mid C_1, \ldots, C_k, C)$ is given above (in Eqn. 1) and the weights given above (in Eqn. 1) are the weights in our current model M together with an initial guess for weight w for condition

C (or the current weight w for condition C if condition C is already in the model). Node 110 may then use the derivatives to find an estimated weight w in a conventional manner using a technique, such as Newton's method. Alternatively, weight w for condition C may be estimated using a random guess, rather than Newton's method.

[0041] Node 110 may then generate a request for statistics that node 110 may send to the other nodes 110 (act 330). The request, in this case, may include the list of features that condition C contains, an identifier corresponding to node 110, and the estimate of the weight determined by node 110. Node 110 may broadcast this request to the other nodes 110.

[0042] Each of nodes 110 receiving the request (hereinafter "receiving nodes") may generate statistics for instances that satisfy condition C (act 340). For example, a receiving node may use its feature-to-instance index to identify the set of instances (within its subset of instances for which it is responsible) that correspond to the features of condition C. Using this set of instances and the current probability of $Y = y_1$ for each of these instances, the receiving node may generate statistics (e.g., derivatives), as described above with respect to Eqn. 2. The receiving nodes may then send the statistics to node 110 that sent the request.

[0043] Node 110 may collect statistics from the receiving nodes and use these statistics to determine a better weight w for condition C (acts 350 and 360). For example, node 110 may use Newton's method to determine a new weight w' from the derivatives generated by the receiving nodes. Node 110 may then use this weight w' to form a rule or update an existing rule: (C, w') (act 370).

[0044] Node 110 may selectively add the rule to the model M (e.g., add a new rule or update an existing rule in the model M) (act 380). To determine whether to add the rule, node 110 may compare the likelihood of the training data D between the current model with the rule (C, w') and the current model without the rule (i.e., $P(D \mid M, (C, w'))$ vs. $P(D \mid M)$). If $P(D \mid M, (C, w'))$ is sufficiently greater than $P(D \mid M)$, then the rule (C, w') may be added to the model M. A penalty or "Cost" for each condition C may be used to aid in the determination of whether $P(D \mid M, (C, w'))$ is sufficiently greater than $P(D \mid M)$. For example, if condition C includes many features, or if the features of condition C are quite rare (e.g., "does the word 'mahogany' appear in the message"), then the cost of condition C could be high. The rule (C, w') may then be added to the model M if: $\text{Log}\ \{P(D \mid M, (C, w'))\} - \text{Log}\{P(D \mid M)\} > \text{Cost}(C)$. If $P(D \mid M, (C, w'))$ is not sufficiently greater than $P(D \mid M)$, then the rule (C, w') may be discarded (i.e., not added to the model M), possibly by changing its weight to zero.

[0045] Node 110 may send the rule to the other nodes 110 (e.g., the receiving nodes) (act 390). If node 110 determined that the rule should not be added to the model M, then node 110 may set the weight for the rule to zero and transmit it to the receiving nodes. Alternatively, node 110 may not send the rule at all when the rule is not added to the model or the rule's weight has not changed. The receiving nodes may use the rule to update their copy of the model, as necessary, and update the current probabilities of $Y = y_1$ for the instances that satisfy the condition contained in the rule (i.e., condition C). The receiving nodes may identify these instances using their feature-to-instance indexes.

[0046] Processing may then return to act 310, where node 110 selects the next candidate condition. Processing may continue for a predetermined number of iterations or until all candidate conditions have been considered. During this processing, each condition may eventually be selected only once or, alternatively, conditions may eventually be selected multiple times.

[0047] As described previously, the acts described with respect to Fig. 3 may occur on multiple nodes 110 concurrently. In other words, various nodes 110 may be sending out statistics requests and processing requests at the same time. It is not necessary, however, that each of nodes 110 perform all of the acts described with regard to Fig. 3. For example, a subset of nodes 110 may select candidate conditions and form rules for the model. The remaining nodes 110 may process the statistics requests, but form no rules.

[0048] Fig. 4 is a flowchart of exemplary processing for generating a model according to a second implementation consistent with the principles of the invention. This processing may also be performed by a combination of nodes 110. Each node 110 may include a copy of the model M and a subset of instances with a current probability of $Y = y_1$ for each instance. Each node 110 may build its own feature-to-instance index for its subset of instances.

[0049] Processing may begin with an empty model M that includes the prior probability of the label. A node 110 may select a candidate condition C to be tested (act 410). It may be possible for multiple nodes 110, or all of nodes 110, to concurrently select candidate conditions. Candidate conditions may be selected in a manner similar to that described above with regard to Fig. 3.

[0050] Node 110 may then generate a request for statistics that node 110 may send to the other nodes 110 (act 420). The request, in this case, may include the list of features that condition C contains and an identifier corresponding to node 110. Node 110 may broadcast this request to the other nodes 110.

[0051] Each of nodes 110 receiving the request (hereinafter "receiving nodes") may generate statistics for instances that satisfy condition C (act 430). For example, a receiving node may use its feature-to-instance index to identify the set of instances (within its subset of instances for which it is responsible) that correspond to the features of condition C. The receiving node may create a histogram of Log P($Y = y_0 \mid C_1, \ldots, C_k$) for the different instances d that satisfy condition C and are labeled y_0, and create another histogram of Log P($Y = y_1 \mid C_1, \ldots, C_k$) for the different instances d that satisfy condition C and are labeled y_1. The receiving nodes may then send the statistics to node 110 that sent the request.

[0052] Node 110 may collect statistics from the receiving nodes and use these statistics to determine a weight w for condition C (acts 440 and 450). For example, node 110 may determine an estimate of weight w from: Sum_d { Log P(Y= y[d] | $C_1, \ldots, C_k$, C) }. Node 110 may then continue to estimate the weight w (e.g., using a binary search, a hill climbing search, or a Newton iteration) until Sum_d { Log P(Y= y[d] | $C_1, \ldots, C_k$, C) } is maximized. Node 110 may then use this weight w to form a rule or update an existing rule: (C, w) (act 460).

[0053] Node 110 may selectively add the rule to the model M (e.g., add a new rule or update an existing rule in the model M) (act 470). To determine whether to add the rule, node 110 may compare the likelihood of the training data D between the current model with the rule (C, w) and the current model without the rule (i.e., P(D | M, (C, w)) vs. P(D | M)). If P(D | M, (C, w)) is sufficiently greater than P(D | M), then the rule (C, w) may be added to the model M. As described above, a penalty or "Cost" may be associated with each condition C to aid in the determination of whether P(D | M, (C, w)) is sufficiently greater than P(D | M). If P(D | M, (C, w)) is not sufficiently greater than P(D | M), then the rule (C, w) may be discarded (i.e., not added to the model M), possibly by changing its weight to zero.

[0054] Node 110 may send the rule to the other nodes 110 (e.g., the receiving nodes) (act 480). If node 110 determined that the rule should not be added to the model M, then node 110 may set the weight for the rule to zero and transmit it to the receiving nodes. Alternatively, node

110 may not send the rule at all when the rule is not added to the model or the rule's weight has not changed. The receiving nodes may use the rule to update their copy of the model, as necessary, and update the current probabilities of $Y = y_1$ for the instances that satisfy the condition contained in the rule (i.e., condition C). The receiving nodes may identify these instances using their feature-to-instance indexes.

[0055] Processing may then return to act 410, where node 110 selects the next candidate condition. Processing may continue for a predetermined number of iterations or until all candidate conditions have been considered. During this processing, each condition may eventually be selected only once or, alternatively, conditions may be selected multiple times.

[0056] As described previously, the acts described with respect to Fig. 4 may occur on multiple nodes 110 concurrently. In other words, various nodes 110 may be sending out statistics requests and processing requests at the same time. It is not necessary, however, that each of nodes 110 perform all of the acts described with regard to Fig. 4. For example, a subset of nodes 110 may select candidate conditions and form rules for the model. The remaining nodes 110 may process the statistics requests, but form no rules.

[0057] Fig. 5 is a flowchart of exemplary processing for generating a model according to a third implementation consistent with the principles of the invention. This processing may also be performed by a combination of nodes 110. Each node 110 may include a copy of the model M (or a fraction of the model M) and a subset of instances with a current probability of $Y = y_1$ for each instance. In this implementation, nodes 110 do not use a feature-to-instance index.

[0058] Generally, the processing of Fig. 5 may be divided into iterations. Rules may be tested or have their weight optimized once per iteration. Each iteration may be broken into two phases: a candidate rule generation phase and a rule testing and optimization phase. The rule testing and optimization phase may determine the weights for conditions generated in the candidate rule generation phase, and accepts rules into the model if their benefit (e.g., difference in log likelihood) exceeds their cost.

[0059] Processing may begin with the generation of new conditions as candidate conditions to test whether they would make good rules for the model M (act 510). The generation of new conditions may concurrently occur on multiple nodes 110. There are several possible ways of generating candidate conditions. For example, candidate conditions might include all conditions with one feature, all conditions with two features that co-occur in some instance, and all extensions of existing rules by one feature (where the combination is in some instance). As a further optimization, extensions of only those rules added in the last iteration may be used.

[0060] The goal of the candidate rule generation phase is to generate new conditions that match some minimum number of instances. There are a couple of strategies for accomplishing this. For example, conditions that appear multiple times in some fraction of the instances (divided among all of nodes 110 and then summed) may be considered. In this case, each node 110 may count the number of instances (of the subset of instances for which node 110 is responsible) that match the condition and generate (condition, count) pairs. The (condition, count) pairs may be gathered at some node 110 (which may be determined by a rule, such as a hash of the condition) and summed. Conditions with some minimum count value may then be kept as candidate conditions. All other conditions may be dropped.

[0061] Alternatively, conditions that appear a certain number of times on a single node 110 may be considered. In other words, each node 110 may count the number of instances (of the subset of instances for which node 110 is responsible) that match the condition. Conditions with some minimum count value on a single node 110 may be kept as candidate conditions. The candidate conditions may be gathered at some node 110 to facilitate the removal of duplicate conditions.

[0062] Then in the rule testing and optimization phase, the candidate conditions may be distributed to all nodes 110 (act 520). Each node 110 may analyze its share of instances to identify which of the candidate conditions match each instance (act 530). Node 110 may store the matching conditions and instances as (condition, instance number) pairs (act 530). Each node 110 may then sort the (condition, instance number) pairs by condition to form a sorted condition-instance list. From the sorted condition-instance list, all instances that match a particular condition may easily be determined.

[0063] Each node 110 may generate statistics for each of the conditions in the sorted condition-instance list (act 540). For example, a node 110 may collect information regarding predicted label probability from the matching instances and the actual number of observed y_0 labels. In one exemplary implementation, nodes 110 may build a histogram based, at least in part, on the collected information and use the histogram as the statistics relating to the condition. In another exemplary implementation, the statistics may take a different form.

[0064] Each node 110 may then send the statistics relating to the condition to a particular node 110 designated to handle that condition. The particular node 110 may be determined, for example, based on a rule, such as a hash of the condition. Node 110 may collect the statistics relating to the condition from the other nodes 110 (act 550). Node 110 may then determine an optimal weight w for the rule (C, w) and determine whether to add the rule to the model M (acts 560 and 570). Node 110 may use techniques similar to those described above with regard to Figs. 3 and 4 to determine the optimal weight w and determine whether to add the rule to the model M.

[0065] Node 110 may then send the rule to the other nodes 110, or just those nodes 110 that sent statistics (i.e., those nodes 110 with instances that match the condition of the rule) (act 580). If node 110 determined that the rule should not be added to the model M, then node 110 may set the weight for the rule to zero and transmit it to the other nodes 110. Alternatively, node 110 may not send the rule at all when the rule is not added to the model. Nodes 110 that receive the rule may use the rule to update their copy of the model, as necessary, and update the predicted label probabilities for the instances that satisfy the condition contained in the rule.

[0066] The rule testing and optimization phase may continue for a number of iterations or until all rules have been tested. The output of the rule testing and optimization phase is new weights for all existing rules (possibly zero if the rule is to be dropped from the model M) and a list of new rules.

[0067] As described previously, the acts described with respect to Fig. 5 may occur on multiple nodes 110 concurrently. In other words, various nodes 110 may be concurrently selecting candidate conditions and/or testing rules for the model M. It is not necessary, however, that each of nodes 110 perform all of the acts described with regard to Fig. 5. For example, a subset of nodes 110 may be responsible for selecting candidate conditions and/or testing rules for the model.

CONCLUSION

[0068] Systems and methods consistent with the principles of the invention may generate a model from a large data set (e.g., a data set that includes possibly millions of data items) efficiently on multiple nodes.

[0069] The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. For example, while series of acts have been described with regard to Figs. 3-5, the order of the acts may be modified in other implementations consistent with the principles of the invention. Also, non-dependent acts may be performed in parallel. Further, the acts may be modified in other ways. For example, in another exemplary implementation, acts 330-360 of Fig. 3 or acts 420-450 of Fig. 4 may be performed in a loop for a number of iterations to settle on a good weight.

[0070] Also, in the three implementations described with regard to Figs. 3-5, for each instance d, there is no need to compute the probability of y[d] given model M every time a condition that instance d satisfies is tested. Instead, there could be an array that keeps the current probability of instance d being y_0 given the model M, and when a condition C is updated, the probabilities for the instances that match that condition C may be updated. The probabilities for the instances that do not match the condition C may be left unchanged.

[0071] It will also be apparent to one of ordinary skill in the art that aspects of the invention, as described above, may be implemented in many different forms of software, firmware, and hardware in the implementations illustrated in the figures. The actual software code or specialized control hardware used to implement aspects consistent with the present invention is not limiting of the present invention. Thus, the operation and behavior of the aspects were described without reference to the specific software code--it being understood that one of ordinary skill in the art would be able to design software and control hardware to implement the aspects based on the description herein.